Technische Universität München **WS 2016**
Fakultät für Informatik **Exercise Sheet 7**
Dr. Tobias Lasser **December 20, 2016**

Exercises in Image Reconstruction and Visualization using C++

This exercise represents the second part of your project. The goal of this part is to implement tomographic image reconstruction using the simulated X-ray data from the first part of the project.

Make sure that you again add all your code to the **part2** git repository, and commit early and often. We will judge your progress and participation also based on the git commits. Please continue to use the basic Qt user interface sketch from the previous part, all GUI elements for this part should be located in the widget for *Reconstruction*.

Exercise 1 On-the-fly system matrix

Implement an *abstract* base class representing the system matrix B of the normal equation

$$\underbrace{A^\top A}_{=B} x = A^\top b$$

and its Tikhonov regularized counterpart ($\lambda \geq 0$)

$$\underbrace{A^\top A + \lambda I}_{=B} x = A^\top b.$$

Please note that *both* B matrices are symmetric positive definite.

This base class should provide (at least) the following methods:

- *mult:* this represents multiplication of the system matrix with a given vector (in “volume space”). You already performed this operation when you simulated X-ray images.
- *multTrans:* this represents multiplication of the transpose of the system matrix with a given vector (in “measurement space”).

During this assignment, we will implement *two* specializations of this abstract base class, one for normal reconstruction, one for regularized reconstruction.

Now, implement the *first specialization* for normal reconstruction.

- The implementation of *mult* and *multTrans* should use two methods *forwardProj* (representing the multiplication Ax) and *backProj* (representing the multiplication $A^\top x$).
- In order to implement these two *Proj methods:
  - Use the forward projection from the first part to implement the forward projection.
  - Implement the back projection (see the provided slides for details).
  - Make sure you use the ray-tracer from the first part in order to compute the forward/back projection *on-the-fly*! For realistic problem sizes the system matrix does not fit into memory, which is why we compute it on-the-fly. Do **not** store the entire system matrix in memory!

- Make sure your code is efficient, as the forward and back projection will be called many times and are the most computationally intensive part of your code.

Exercise 2 **Tomographic reconstruction**

Using the simulated X-ray data from the first part and the on-the-fly system matrix from the previous exercise, we now compute tomographic reconstructions iteratively.

- Implement the Conjugate Gradient (CG) algorithm (see the provided slides for details and references).
- Use the zero vector as the starting value for your iterations.
- Provide the system matrix to the CG algorithm as a reference/pointer parameter of the type of your abstract base class, so that we can call the CG algorithm with either specialization of the base class.
- Visualize the result of the reconstruction in the Qt GUI somehow. For example, a simple 2D slice-viewer would suffice.

Exercise 3 **Finding good parameters**

In order to get good reconstructions, you have to choose a good set of parameters. In this exercise you will experiment in order to find a good set of parameters, notably:

- The most important set of parameters are the *acquisition poses*, i.e. the positions and orientations of your source/detector pairs. Generate a suitable trajectory of acquisition poses, which cover the volume of interest sufficiently. Play around with the number of acquisition poses as well as the positions and orientations.
- Select a suitable number of iterations for the CG algorithm.
- Provide mechanisms in your Qt GUI to change these parameters (e.g. a spin box for number of iterations, or a drop down box for different trajectories).

After this exercise, your code should include a reasonable set of parameters, such that the provided example phantoms (the .edf volumes) reconstruct perfectly (i.e. your reconstruction result is identical to the volume used for simulation). Make sure your GUI has these reasonable parameters set as default values!

Exercise 4 **Noise**

Unfortunately, the real world is not perfect, meaning tomographic reconstructions will never be perfectly true to the original. In order to be a little bit more realistic, we now want to add noise to our simulated X-ray data.

- Add an additional step at the end of your X-ray data simulation which adds 2% of white Gaussian noise to your previously simulated data y:

  $$y_{\text{noise}} = y + 0.02 \cdot \big(\max(y) - \min(y)\big) \cdot \mathit{randn}$$

  where *randn* denotes a vector of the same size as y, but filled with normally distributed random values.
- Provide a mechanism in the Qt GUI to turn on/off the noise on the simulated X-ray data. Optionally, you can also implement selection of the amount of noise (not just fixed to 2%).

Exercise 5 Regularized reconstruction

In order to deal with the noisy data, we now want to use *Tikhonov* regularization for reconstruction, see the provided slides for an introduction.

- Implement a *second specialization* of the abstract system matrix class. This implementation should reflect the matrix:

  $$A^\top A + \lambda I$$

  where A denotes the standard system matrix, I represents the identity matrix, and $\lambda \in \mathbb{R}$ is the regularization parameter.
- Make sure again that your implementation is on-the-fly and efficient. Avoid code duplication!
- Add a mechanism to the Qt GUI that allows you to choose between normal and regularized reconstruction, and that allows choosing the parameter λ (e.g. using a spin box).
- Visualize the results, just as in exercise 2. Make sure that the results make sense (for example, for $\lambda = 0$ results should be identical to exercise 2).

Exercise 6 Finding good parameters again

The parameter λ strongly influences the quality of the reconstruction. $\lambda = 0$ will degenerate the problem to the standard un-regularized reconstruction, while $\lambda = \infty$ will lead to a *zero* solution. Thus this parameter has to be chosen carefully.

- Play around with λ, find a value that works reasonably well across the provided example volumes. Ideally the parameter is chosen such that the noise is mostly suppressed, but also that the regularization does not dominate the entire image.
- Be aware that the other parameters (the input of the simulation, the trajectory, the number of iterations) and λ influence each other.
- Make sure your Qt GUI facilitates this "playing around", and that you set a reasonable value of λ as the default.